

[illegible]

```

SSSSSSSS DDDDDDDD LL
SSSSSSSS DDDDDDDD LL
SS        DD        DD LL
SS        DD        DD LL
SS        DD        DD LL
SS        DD        DD LL
        SSSSSS DD        LL
        SSSSSS DD        LL
                SS DD        LL
                SS DD        LL
                SS DD        LL
                SS DD        LL
SSSSSSSS DDDDDDDD LLLLLLLLLL
SSSSSSSS DDDDDDDD LLLLLLLLLL

```

{ REQUIRE file for I/O Statement Block (ISB)
{ File: OTSISB.SDL Edit: SBL2005
{

{*****
{
{* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
{* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
{* ALL RIGHTS RESERVED. *
{* *
{* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
{* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
{* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
{* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
{* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
{* TRANSFERRED. *
{* *
{* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
{* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
{* CORPORATION. *
{* *
{* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
{* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
{* *
{*****

{
{ Author: T.Hastings
{ [Previous edit history removed. SBL 19-Aug-1982]
{ ***** - VMS Version V2.0
{ 1-057 - Force length calculation of block after allocation. HJ 22-Aug-1980
{ 1-058 - Add statement types for FORTRAN NAMELIST. Fix some typos in
{ comments. SBL 27-August-1980
{ 1-059 - Add flags ISBSV_SINGL_ELEM and ISBSV_NEED_INIT to support
{ Fortran single-element lists. JAW T1-May-1981
{ 1-060 - Restore names for input format flags, and add output format
{ flag ISBSV_ERR_OFLO. JAW 13-Aug-1981
{ 1-061 - Remove date from Edit: line for uniformity. JAW 15-Aug-1981
{ ***** - VMS Version V3.0
{ 1-062 - Add new statement types for Basic V2. PLL 1-Jun-1982
{ 2-001 - Convert to SDL. SBL 19-Aug-1982
{ 2-002 - Don't depend on names for unions/structures. SBL 29-Sep-1982
{ 2-003 - Change aggregate name to ISB for better fieldset name. SBL 26-Oct-1982
{ 2-004 - Add A_PREVIOUS_LUB. SBL 2-Dec-1982
{ 2-005 - Add A_SAVE_PTR and A_SAVE_END, and statement types for list-directed
{ internal files. SBL 21-Apr-1983
{
{--

{+
{ The ISB contains OTS OWN data associated with a particular
{ logical unit which is needed only for the set of calls which
{ implement a single I/O statement. The ISB locations are grouped by
{ level of abstraction:
{ 1. User Program Interface (UPI)
{ 2. User Data Formatter (UDF)
{ 3. Record Formatter and processor (REC)
{


```
{ In principle, ISB could be dynamically allocated at the beginning of every
{ READ, WRITE, DECODE, and ENCODE and deallocated at the end
{ (in FOR$IO_END). However, for runtime efficiency it is not.
{ OWN data which is needed to be retained for more than
{ one I/O Statement is allocated in the Logical Unit Block (LUB).
{ Data which is needed during a single call is always LOCAL.
{
{
```

```
{ ISB definition (-11 OTS equivalents are indicated in parens)
{ All are unsigned, except ISB$W_FMT_REP.
{ Symbols have form: ISB$t_symbol where t is A,B,W,L,T, or V.
{-
```

```
MODULE $ISBDEF;
AGGREGATE ISB STRUCTURE PREFIX ISB$ ORIGIN end_of_lub;
```

```
{+
{ I/O statement type codes
{ Used to index into routine dispatch tables to call
{ the appropriate User-program data formatter level
{ of abstraction (UDF = level 2) and Record processing level
{ of abstraction (REC = level 3).
{ Codes assigned so that a TRUE value means WRITE and a
{ false value means READ. The distinction between
{ formatted and object-time formatted disappears at level 1
{ and so does not have a separate statement type code.
{-
```

```
{ the 0 entry is no longer used because it reports an error. It is
{ designed to catch a recursive CLOSE among other things.
```

CONSTANT ST_TY_WSF	EQUALS 1;	{ FORTRAN WRITE sequential formatted
CONSTANT FORSTTYLO	EQUALS 1;	{ Lowest FORTRAN statement type
CONSTANT ST_TY_RSF	EQUALS 2;	{ FORTRAN READ sequential formatted
CONSTANT ST_TY_WSU	EQUALS 3;	{ FORTRAN WRITE sequential unformatted
CONSTANT ST_TY_RSU	EQUALS 4;	{ FORTRAN READ sequential unformatted
CONSTANT ST_TY_WDF	EQUALS 5;	{ FORTRAN WRITE direct formatted
CONSTANT ST_TY_RDF	EQUALS 6;	{ FORTRAN READ direct formatted
CONSTANT ST_TY_WDU	EQUALS 7;	{ FORTRAN WRITE direct unformatted
CONSTANT ST_TY_RDU	EQUALS 8;	{ FORTRAN READ direct unformatted
CONSTANT ST_TY_WSL	EQUALS 9;	{ FORTRAN WRITE sequential list-directed
CONSTANT ST_TY_RSL	EQUALS 10;	{ FORTRAN READ sequential list-directed
CONSTANT ST_TY_WMF	EQUALS 11;	{ FORTRAN WRITE memory formatted (ENCODE)
CONSTANT MIN_DE_EN	EQUALS 11;	{ FORTRAN Minimum
		{ DECODE/ENCODE code
CONSTANT ST_TY_RMF	EQUALS 12;	{ FORTRAN READ memory formatted (DECODE)
CONSTANT MAX_DE_EN	EQUALS 12;	{ FORTRAN Maximum
		{ DECODE/ENCODE code
CONSTANT ST_TY_WXF	EQUALS 13;	{ FORTRAN REWRITE indexed formatted
CONSTANT ST_TY_RKF	EQUALS 14;	{ FORTRAN READ keyed formatted
CONSTANT ST_TY_WXU	EQUALS 15;	{ FORTRAN REWRITE indexed unformatted
CONSTANT ST_TY_RKU	EQUALS 16;	{ FORTRAN READ keyed unformatted
CONSTANT ST_TY_WIF	EQUALS 17;	{ FORTRAN WRITE internal formatted
CONSTANT ST_TY_RIF	EQUALS 18;	{ FORTRAN READ internal formatted
CONSTANT ST_TY_WSN	EQUALS 19;	{ FORTRAN WRITE sequential NAMELIST
CONSTANT ST_TY_RSN	EQUALS 20;	{ FORTRAN READ sequential NAMELIST

CONSTANT ST_TY_WIL	EQUALS 21:	{ FORTRAN WRITE internal list-directed
CONSTANT ST_TY_RIL	EQUALS 22:	{ FORTRAN READ internal list-directed
CONSTANT FORSTTYHI	EQUALS 22:	{ Highest FORTRAN statement type
		{ Leave a little room for FORTRAN expansion
CONSTANT ST_TY_PRI	EQUALS 27:	{ BASIC PRINT
CONSTANT BASSTTYLO	EQUALS 27:	{ Lowest BASIC statement type
CONSTANT ST_TY_LIN	EQUALS 28:	{ Basic LINPUT
CONSTANT ST_TY_PSE	EQUALS 29:	{ Basic PUT sequential
CONSTANT ST_TY_INP	EQUALS 30:	{ Basic INPUT
CONSTANT ST_TY-PRU	EQUALS 31:	{ Basic PRINT USING
CONSTANT ST_TY-INL	EQUALS 32:	{ Basic INPUT LINE
CONSTANT ST_TY-DEL	EQUALS 33:	{ Basic DELETE
CONSTANT ST_TY-REA	EQUALS 34:	{ Basic READ memory
CONSTANT ST_TY-UPD	EQUALS 35:	{ Basic UPDATE
CONSTANT ST_TY-GSE	EQUALS 36:	{ Basic GET sequential
CONSTANT ST_TY-RES	EQUALS 37:	{ Basic RESTORE
CONSTANT ST_TY-SCR	EQUALS 38:	{ Basic SCRATCH
CONSTANT ST_TY-PRE	EQUALS 39:	{ Basic PUT relative
CONSTANT ST_TY-GRE	EQUALS 40:	{ Basic GET relative
CONSTANT ST_TY-FRE	EQUALS 41:	{ Basic FIND relative
CONSTANT ST_TY-UNL	EQUALS 42:	{ Basic UNLOCK
CONSTANT ST_TY-FEE	EQUALS 43:	{ Basic FREE (strange name to avoid conflict
		{ with FIND relative)
CONSTANT ST_TY_GIN	EQUALS 44:	{ Basic GET indexed
CONSTANT ST_TY-PIN	EQUALS 45:	{ Basic Put indexed
CONSTANT ST_TY-MOV	EQUALS 46:	{ BASIC MOVE FROM/MOVE TO
CONSTANT ST_TY-FIN	EQUALS 47:	{ Basic FIND indexed
CONSTANT ST_TY-MIN	EQUALS 48:	{ Basic MAT INPUT
CONSTANT ST_TY-RIN	EQUALS 49:	{ Basic RESTORE indexed
CONSTANT ST_TY-MLI	EQUALS 50:	{ Basic MAT LINPUT
CONSTANT ST_TY-FSE	EQUALS 51:	{ Basic FIND sequential
CONSTANT ST_TY-MPR	EQUALS 53:	{ Basic MAT PRINT
CONSTANT ST_TY-MRE	EQUALS 54:	{ Basic MAT READ
CONSTANT ST_TY-GRFA	EQUALS 55:	{ Basic GET by RFA
CONSTANT ST_TY-FRFA	EQUALS 56:	{ Basic FIND by RFA
CONSTANT BASSTTYHI	EQUALS 56:	{ Highest BASIC statement type

{ end of statement type definitions

{ Begin data structure definition

```
union 1 UNION;
  RESTARTPC ADDRESS;      { Address of start of I/O list, for
                           { restarting BASIC I/O statements.
  USR_HANDL ADDRESS;      { Address of user's handler (FORTRAN)
END union_1;
```

```
union 1A UNION;
  MAJ_F_PTR ADDRESS;      { Holds pointer to last Basic major frame.
  PREVIOUS_LUB ADDRESS;   { Back pointer to previous LUB (FORTRAN)
END union_1A;
```

```
USER_FP ADDRESS;          { User's FP.
union 1B UNION;
  FRT_STKP WORD UNSIGNED DIMENSION 8; { 8 entry pushdown stack containing relative
```



```

      { byte offset in format statement for beginning
      { of a repeat group. ISB$B_FMT_DEP is index
      { into stack (-1 = empty, 0 = 1 item, 1 = 2 items,...)
{ The following two items are used by FOR$UDF_RL to save the buffer
{ pointer and end when processing a repeated complex value.

```

```

structure_1B STRUCTURE;
  SAVE_PTR ADDRESS;      { Saved LUB$A_BUF_PTR
  SAVE_END ADDRESS;      { Saved LUB$A_BUF_END
end structure_1B;
end union_1B;

```

```

union_2 UNION;
  FMT_STKR WORD UNSIGNED DIMENSION 8; { 8 entry pushdown stack containing group repeat
      { count (as a word) remaining. ISB$B_FMT_DEP is
      { index into stack (-1=empty, 0=1 item, 1=2
      { items, ...)
  SCA_FAC_D BYTE UNSIGNED DIMENSION 8; { Double precision scale factor for BASIC
END union_2;

```

```

{+
{ Locations initialized for all I/O statements
{-

```

```

union_3 UNION;
  ERR_NO BYTE UNSIGNED;      { FORTRAN error number occurring during
      { current I/O statement and continued until
      { end of statement where it will be SIGNAled.
      { 0 means no such continuable error has occurred in this
      { I/O statement.
  SCALE_FAC BYTE;           { BASIC scale factor in the range of -6 -> 0.
END union_3;

```

```

{+
{ ISB Locations set at the beginning of every I/O statement in Procedures
{ at the User Program Interface (FOR-UPI) level of abstraction, which is:
{ FOR$(READ,WRITE)_{SF,SO,SU,DF,DO,DU} or FOR$(DECODE,ENCODE)_{MF,MO}
{-

```

```

STTM_TYPE BYTE UNSIGNED;    { (FOR-RECIO,W.EXJ) Record I/O statement
      { type code. Used as an index into
      { dispatch table structures for calling
      { procedures in the User Data Formatter (FOR-UDF)
      { and Record processing (FOR-REC) levels
      { of abstraction. See FOR$IO_BEG Module.
FMT_LEN WORD UNSIGNED;      { (FORFMTBUF) No. of characters
      { allocated to contain compiled format of
      { object-time format. 0 means not
      { object-time format. Space is deallocated
      { at end of I/O statement (FORFOR$IO_END).
ERR_EQUAL ADDRESS;          { (FOR-ERREX) Adr of ERR= transfer or 0 if none.
END_EQUAL ADDRESS;          { (FOR-ENDEX) Adr of END= transfer or 0 if none.
FMT_BEG ADDRESS;            { Address of the beginning of the FORTRAN format.
      { This is set in FOR$IO_BEG and can either point
      { to a precompiled format or a run-time compiled
      { format. In the latter case, FMT_LEN is non-zero.
      { This is also where the NAMELIST description

```

{ block address is stored.

```
{+
{ ISB locations used by the I/O independent format interpreter
{ FOR$$FMT_INTRP and occasionally updated by the input or output
{ dependent Formatted User Data Formatter (FOR$$UDF_RF or FOR$$UDF_WF)
{ for Hollerith (FOR-nH) format code only.
{-
```

```
union 3A UNION;
  FMT_PTR ADDRESS;          { (FORFMTAD) Adr. of next byte to be read
                             { from the compiled format statement byte array
  LIS_HEAP_LEN LONGWORD UNSIGNED; { Length of storage allocated in LIS_STR
                             { Used by FOR$$UDF_RL
  END union_3A;
```

```
{+
{ ISB locations used as own storage solely by FOR$$UDF_RL
{ the list-directed input processor. More storage is defined
{ further down where it is convenient.
{-
```

```
LIS_STR ADDRESS;          { Address of repeated string constant
                           { saved in FOR$$UDF_RL1.
```

```
{+
{ ISB Locations returned as parameters from FOR$$FMT_INTRP to the input
{ or output dependent Formatted User Data Formatter (FORFOR$$UDF_RF
{ or FOR$$UDF_WF) which do not modify them. These parameters are
{ stored in the ISB because they are needed by FOR$$FMT_INTRP for more than
{ one call if the format code is repeated.
{-
```

```
FMT_P BYTE;              { (FOR-PSCALE) Signed P scale factor
FMT_W WORD UNSIGNED;      { (FOR-W) Width of field in characters
FMT_D BYTE UNSIGNED;      { (FOR-D) Number of fraction digits
FMT_E BYTE UNSIGNED;      { (FOR-E) Number of exponent characters
```

```
{+
{ ISB Locations used solely by the I/O independent format interpreter
{ FOR$$FMT_INTRP(0,1)
{-
```

```
union 4 UNION;
  FMT_REP WORD;           { (FOR-REPCNT) signed format repeat count for current
                           { format code.
  LIS_REP WORD UNSIGNED;  { (FOR-REPCT) unsigned repeat count for List-
                           { directed input
  LEN_REM WORD UNSIGNED;  { (BAS-new) length of format string remaining.
  END union_4;
```

```
union 5 UNION;
  FMT_CODE UNION UNION;
  FMT_CODE BYTE UNSIGNED; { Format type code
```

```

    FMT_CODE_STRUCT STRUCTURE;
    -fill_3 BITFIELD LENGTH 7 FILL TAG $$; { first 7 bits are format code
    FMT_REPR BITFIELD; { representation byte follows if 1
    END FMT_CODE_STRUCT;
    END FMT_CODE_UNION;
    LIS_CTYPE BYTE UNSIGNED; { type of constant scanned by list-directed input
    END union_5;

    FMT_REVER WORD UNSIGNED; { (FORFMTAD) Relative position of current format reversion
    { point to revert to when end of format
    { statement is encountered with more data
    { elements to be transmitted.
    FMT_DEP BYTE UNSIGNED; { (FORFSTKP) Adr. of current top of format
    { pushdown stack.

    FMT_FLAGS_UNION UNION;
    -FMT_FLAGS WORD UNSIGNED; { Flags for FORTRAN-77
    FMT_FLAGS_STRUCT STRUCTURE;
    -INP_FLAGS_UNION UNION;
    -INP_FLAGS_BYTE UNSIGNED; { Input conversion flags
    INP_FLAGS_STRUCT STRUCTURE;
    -BN BITFIELD; { Blanks are nulls if set
    ONLY_E BITFIELD; { Only allow E, e if set
    ERR_OFLO BITFIELD; { Underflow is an error if set
    DONTROUND BITFIELD; { Don't round result if set
    SKIPTABS BITFIELD; { Ignore tabs if set
    EXP_LETTER BITFIELD; { Exponent letter is required if set
    FORCESCALE BITFIELD; { Scale even if exponent present if set
    fill_4 BITFIELD FILL TAG $$; { Expansion
    END INP_FLAGS_STRUCT;
    END INP_FLAGS_UNION;
    OUT_FLAGS_UNION UNION;
    -OUT_FLAGS_BYTE UNSIGNED; { Output conversion flags
    OUT_FLAGS_STRUCT STRUCTURE;
    -SP BITFIELD; { Force optional +
    ERR_OFLO BITFIELD; { Exponent field width overflow is an
    { error if set
    fill_5 BITFIELD LENGTH 6 FILL TAG $$; { Expansion
    END OUT_FLAGS_STRUCT;
    END OUT_FLAGS_UNION;
    END FMT_FLAGS_STRUCT;
    END FMT_FLAGS_UNION;

    fill_6 BYTE UNSIGNED FILL TAG $$; { 1 spare byte for future (FOR-keep longword aligned)

    (+
    { Status bits used at any of the levels of abstraction
    (-

    STTM_STAT_UNION UNION;
    STTM_STAT_WORD UNSIGNED; { status lasting only for a single
    { I/O statement, but needed
    { across several calls which
    { implement that single I/O statement.

    STTM_STAT_STRUCTURE STRUCTURE;

```



```

P_FORM_CH BITFIELD LENGTH 2: { (BAS) Store the format character that follows
                                { Prompt. This is set in BAS$$UDF_RL1 and read
                                { in BAS$IO_END.
DOLLAR BITFIELD: { (FOR-DOLFLG) Dollar format encountered in
USER_ELEM BITFIELD: { format processing
                                { (FORFMTLP) User-program data encountered
                                { in format for current records. Used to
                                { group with no data element format code
                                { thereby causing an infinite loop
SLASH BITFIELD: { (FOR-W.NULL) Slash seen during formatted input.
                                { *** also: Slash seen in List-directed input.
LAST_REC BITFIELD: { (FOR-UNFLGS) Last record in segmented record
DE_ENCODE BITFIELD: { being processing if 1, 0=not last record
                                { (FOR-DV.FAK) DECODE/ENCODE being done
                                { so RAB and unit number have
                                { no meaning (FOR-used during error handling).
                                { Also set for internal files.
LIS_HEAP BITFIELD: { List directed input currently has heap
RECURSIVE BITFIELD: { storage allocated.
                                { Used by OTSCCB for recursive I/O.
MAT_CONT BITFIELD: { Set when there is I/O in progress for
                                { this LUN in addition to the current I/O.
MAT_PRINT BITFIELD: { (BAS) MAT INPUT continuation - "&" was last
                                { character of record. Read and written at the
                                { REC level of matrix processing.
                                { (BAS) MAT PRINT has more than one array for an
                                { element transmitter. Set and checked in UPI
                                { level of MAT INPUT element transmitter.
                                { Cleared by IO_END.
PRINT_INI BITFIELD: { (BAS) A print statement has been initialized.
                                { set in BASIOBEG. cleared in element transmitter
                                { checked and cleared in BASIOEND. Used to indicate
                                { that there has been a PRINT with no element
                                { transmitter.
SNGL_ELEM BITFIELD: { (FOR) There is only one element in the
                                { current I/O list. Indicates that an
                                { unbuffered transfer is possible if
                                { record type and record size permit.
NEED_INIT BITFIELD: { (FOR) REC-level initialization has not
                                { yet been done. Set on an unformatted
                                { READ, other than a keyed READ.

                                END STTM_STAT_STRUCTURE;
                                END STTM_STAT_UNION;

INTFILEND ADDRESS: { End of internal file buffer (FORTRAN)

CONSTANT NEG_LUB EQUALS .: { Negative length of LUB (which follows)

{+
{ The following filler occupies the space where the LUB is allocated.
{ If the length of the LUB is changed the size of this filler must be
{ changed accordingly.
{-
lub_filler BYTE UNSIGNED DIMENSION 100 FILL TAG $$;

```

OTSISB.SDL:1

16-SEP-1984 16:41:38.57^{M 13} Page 8

CONSTANT ISB_LEN EQUALS ::

end_of_lub BYTE FILL TAG \$\$;
END ISB;

END_MODULE \$ISBDEF;

{ End of file OTSISB.SDL

★★

0202 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

BRARM5G
LIS

LIBRTL

LIBRTL2
MAP

LIBFMTDEF
SDL

LIBPROLOG
REQ

LIBRTL
MAP

OTSLB
SDL

SUBS
LIS

OTSCBREQ
REQ

OTSMAC
REQ

LISTLIB
LIS

LIBCLIDEF
SDL

LIBLNK
REQ

OTSLNK
REQ

LIBOCFDEF
SDL

RTLILB
REQ

PROCMD
LIS

LIBMACROS
REQ

OTSLB
SDL